# Design Patterns Used in Spring

This repository is a simple spring boot application, that demonstrates a few design patterns:

* Singleton
* Controller
* Factory
* Strategy
* Proxy
* Observer
* Aspect-oriented programming

# Strategies for Building Robust Security

* Unified Process for Security Design
* Design Patterns
* Best Practices
* Reality Checks
* Proactive Assessment
* Profiling
* Defensive Strategies
* Recovery and Continuity Strategie

# The Importance of Security Compliance

* Sarbanes-Oxley Act
* Gramm-Leach-Bliley Act
* HIPPA
* The Children’s Online Privacy Protection Act
* EU Directive on Data Protection
* California’s Notice of Security Breach (1798.29)
* Security Compliance in Other Countries

# The Importance of Identity Management

* Identity Provisioning Services
* Identity Data Synchronization Services
* Access Management Services
* Federation Services
* Directory Services
* Auditing and Reporting Services

# Secure Personal Identification

* Smart Card Identity
* Biometric Identity
* RFID-Based Identity

# Security Requirements and Goals

* Confidentiality
* Integrity
* Authentication
* Authorization
* Non-repudiation
* The Role of Cryptography in Security
* Cryptographic Algorithms
* The Role of Secure Sockets Layer (SSL)
* Threat Modeling

# Identity Management

* Single Sign-on (SSO)
* Federated SSO

# Java Security Management tools

* Java Keystore
* Keytool
* Policytool
* Jarsigner

# J2EE Container-Based Security

* Declarative Security
* Programmatic Security
* J2EE Authentication
* Protection Domains
* J2EE Authorization
* Java Authorization Contract for Client Containers (JACC)
* Transport Layer Security

# J2EE Component/Tier-Level Security

Users, Groups, Roles, and Realms

# Web- or Presentation-Tier Security

* Web tier authentication mechanisms
* Using JAAS for Web-tier Authentication
* Using Agent based Web-tier Authentication
* Single Sign-On Authentication for Web Applications
* HTTP Session Tracking, Cookies and URL Rewriting
* Web-Tier Authorization mechanisms

# Web Services Security Requirements

* Authentication
* Authorization and Entitlement
* Auditability and Traceability
* Data Integrity
* Data Confidentiality
* Non-repudiation
* Availability and Service Continuity
* Single Sign-on and Delegation
* Identity and Policy Management
* Security Interoperability

# Security Testing

* Black Box Testing
* White Box Testing

# Securing the Web Tier: Design Strategies and Best Practices

# Web-tier Security Patterns

* Authentication Enforcer
* Authorization Enforcer
* Intercepting Validator
* Secure Base Action
* Secure Logger
* Secure Pipe
* Secure Service Proxy
* Intercepting Web Agent

# Securing the Business Tier: Design Strategies and Best Practices

# Security Considerations in the Business Tier

# Business Tier Security Patterns

* Audit Interceptor
* Container Managed Security
* Dynamic Service Management
* Obfuscated Transfer Object
* Policy Delegate
* Secure Service Façade
* Secure Session Object

# Securing Web Services: Design Strategies and Best Practices

# Web Services Security Protocols Stack

* Network-Layer Security
* Transport-Layer Security
* Message-Layer Security

# Web Services Security Infrastructure

* Network Perimeter Security
* XML Firewall
* Web Services Infrastructure
* Identity Provider
* Directory Services

# Web Services Security Patterns

* Message Interceptor Gateway
* Message Inspector
* Secure Message Router

# Securing the Identity: Design Strategies and Best Practices

# Identity Management Security Patterns

* Assertion Builder Pattern
* Single Sign-on (SSO) Delegator Pattern
* Credential Tokenizer Pattern

**Building End-to-End Security Architecture: A Case Study**

Overview

*Understanding the Security Challenges*

*Assumptions*

Use Case Scenarios

Choosing the Right Methodology

Identifying the Requirements

*Identifying the Security Requirements*

*System Constraints*

*Security Use Cases*

*System Environment*

Application Architecture

*Conceptual Security Model*

Security Architecture

* *Risk Analysis and Mitigation*
* *Trade-Off Analysis (TOA)*
* *Applying Security Patterns*
* *Security Architecture – Detailed Components*

Design

*Policy Design*

*Factor Analysis*

*Security Infrastructure*

*Tier Analysis*

*Trust Model*

*Threat Profiling*

*Security Design*

Development

*Unit and Integration Testing*

Testing

* *White Box Testing*
* *Black Box Testing*

Deployment

* *Configuration*
* *Monitoring*
* *Auditing*

# **Big endian and little endian**

## Writer

[](https://github.com/tdm1223) [](https://github.com/rlatjdwo555) [](https://github.com/Stupid07)

## Endianness

* It refers to a method of arranging several consecutive objects in a one-dimensional space such as a computer's memory.
* The way bytes are arranged is called byte order **.**
* 빅엔디언: The larger unit comes first (MSB [1)](https://github.com/jobhope/TechnicalNote/blob/master/computer_architecture/BigEndianAndLittleEndian.md#ref1) )
* 리틀엔디언: the smaller unit comes first (LSB [2)](https://github.com/jobhope/TechnicalNote/blob/master/computer_architecture/BigEndianAndLittleEndian.md#ref1) )
* 미들엔디언Note: Some architectures use different orderings for 2-byte units and 1-byte units.

## Big-endian and little-endian in byte order

| **type** | **Representation of 0x1234** | **Representation of 0x12345678** |
| --- | --- | --- |
| big endian | 12 34 | 12 34 56 78 |
| little endian | 34 12 | 78 56 34 12 |
| middle endian | - | 34 12 78 56 or 56 78 12 34 |

* Most computers use **little-endian** , and networks use addresses in **big-endian .**
* Due to the influence of the network, many protocols and files are composed of **big-endian methods.**
* **Big endian** makes debugging easier because it is the same as how humans read numbers, and it has an advantage over little **endian in comparison operations because the large unit comes first.**
* **Little-endian** has an advantage over **big-endian** in arithmetic operations because it can process small values ​​first .

## reference

* [endianness wiki](https://ko.wikipedia.org/wiki/%EC%97%94%EB%94%94%EC%96%B8)

## footnote

### 1) MSB(Most Significant Byte)

* How to write the largest byte to the front (or left)
* Lowercase msb usually means bit.

### 2) LSB(Least Significant Byte)

* How to write the smallest byte to the front (or left)
* Lowercase lsb usually means bit.